IN THE SPECIFICATION:

(1) Please replace paragraph 31 with the following paragraph showing changes.

More precisely, the table compressor ~~compression system~~ 100 generally selects a certain subset of attributes (referred to as "predicted" attributes) for which no values are explicitly stored in the compressed table; instead, concise CaRTs that predict these values (within the prescribed error bounds) are maintained. Thus, for a predicted attribute X that is strongly correlated with other attributes in the table, the table compression system 100 is typically able to obtain a very succinct CaRT predictor for the values of X, which can then be used to completely eliminate the column for X in the compressed table. Clearly, storing a compact CaRT model in lieu of millions or billions of actual attribute values can result in substantial savings in storage. In addition, allowing for errors in the attribute values predicted by a CaRT model only serves to reduce the size of the model even further and, thus, improve the quality of compression; this is because, as is well known to those skilled in the art, the size of a CaRT model is typically inversely correlated to the accuracy with which it models a given set of values.

(2) Please replace paragraph 41 with the following paragraph showing changes.

One algorithmic issue faced by the table compressor ~~compression system~~ 100 is that of computing an optimal set of CaRT models for the input table such that (a) the overall storage requirements of the compressed table are minimized, and (b) all predicted attribute values are within the user-specified error bounds. This can be a very challenging optimization problem since, not only is there an exponential number of possible CaRT-based models to choose from, but also building CaRTs (to estimate their compression benefits) is a computation-intensive task, typically requiring multiple passes over the data. As a consequence, the table compressor 100 employs a number of sophisticated techniques from the areas of knowledge discovery and combinatorial optimization in order to efficiently discover a "good" (sub)set of predicted attributes and construct the corresponding CaRT models.

(3) Please replace paragraph 47 with the following paragraph showing changes.

Abstractly, the semantic compression algorithms seek to partition the set of input attributes X into a set of *predicted attributes* $\{X_1,\ldots,X_p\}$ and a set of *predictor attributes* $X - \{X_1,\ldots,X_p\}$ such that the values of each predicted attribute can be obtained within the specified error bounds based on (a subset of) the predictor attributes through a small classification or regression tree (except perhaps for a small set of outlier values). (The notation $X_i \rightarrow X_i$ is used to denote a CaRT predictor for attribute X_i using the set of predictor attributes $\underline{X_i - \{X_1,\ldots,X_p\}}$ ~~$X_i \subseteq X - \{X_1,\ldots,X_p\}$~~.) Note that we do not allow a predicted attribute X_i to also be a predictor for a different attribute. This restriction is important since predicted values of X_i can contain errors, and these errors can cascade further if the erroneous predicated values are used as predictors, ultimately causing error constraints to be violated. The final goal, of course, is to minimize the overall storage cost of the compressed table. This storage cost $[T_c]$ is the sum of two basic components:

1. *Materialization cost, i.e.*, the cost of storing the values for all predictor attributes $X - \{X_1,\ldots,X_p\}$. This cost is represented in the T^1 component of the compressed table, which is basically the projection of T onto the set of predictor attributes. (The storage cost of materializing attribute X_1 is denoted by a procedure MaterCost (X_i).)
2. *Prediction Cost, i.e.* the cost of storing the CaRT models used for prediction plus (possibly) a small set of outlier values of the predicted attribute for each model. (The storage cost of predicting attribute X_i through the CaRT predictor $X_i \rightarrow X_i$ is denoted by a procedure

PredCost($X_i \rightarrow X_i$); note that this does *not* include the cost of materializing the predictor attributes in X_i.)

(4) Please replace paragraph 49 with the following paragraph showing changes.

Turning once again to FIGURE 1, disclosed is one embodiment of the table compressor ~~compression system~~ 100 that includes four major functional blocks: a DependencyFinder 110, a CaRTSelector 120, a CaRTBuilder 130 and a RowAggregator 140. In the following paragraphs, a brief overview of each functional block is provided; a more detailed description of each functional block and the relevant algorithms are discussed more fully below.

(5) Please replace paragraph 50 with the following paragraph showing changes.

Generally, the DependencyFinder 110 produces a mathematical model, also known as an interaction model, using attributes of a data table 112. The data output of the interaction model are then used to guide CaRT building algorithms, such as those used by the CaRTSelector 120 and the CaRTBuilder 130 ~~140~~. The DependencyFinder 110 builds this interaction model in part because there are an exponential number of possibilities for building CaRT-based attribute predictors, and therefore a concise model that identifies the strongest correlations and "predictive" relationships in input data is needed. To help determine these correlations and "predictive" relationships in input data, an approach used by the DependencyFinder 110 is to construct a Bayesian network 115 model that captures the statistical interaction of an underlying set of attributes *X*.

(6) Please replace paragraph 52 with the following paragraph showing changes.

After the Bayesian network 115, has been built, the CaRTSelector 120 will then be executed. Given the input table T 112 and error tolerances e_i 123, (as well as the Bayesian network 115 on the attributes of T built by the DependencyFinder 110,) the CaRTSelector 120 is generally responsible for selecting a collection of predicted attributes and the corresponding CaRT-based predictors such that a final overall storage cost is minimized (within the given error bounds). The CaRTSelector 120 employs the Bayesian network 115 built on X to intelligently guide a search through the huge space of possible attribute prediction strategies. Clearly, this search involves repeated interactions with CaRTBuilder 130 ~~110~~ which is responsible for actually building the CaRT-models for the predictors.

(7) Please replace paragraph 71 with the following paragraph showing changes.

The CaRTSelector 120 is an integral part of the table compressor 100 model-based semantic compression engine. Given the input data table and error tolerances, as well as the Bayesian network capturing the attribute interactions, a goal of the CaRTSelector 120 is to select (a) a subset of attributes to be predicted and (2) the corresponding CaRT-based predictors, such that the overall storage cost is minimized within the specified error bounds. As discussed above, the total storage cost T_c is the sum of the materialization costs (of predictor attributes) and prediction costs (of the CaRTs for predicted attributes). In essence, the CaRTSelector 120 implements the core algorithmic strategies for solving the CaRT-based semantic compression problem. Deciding on a storage-optimal set of predicted attributes and corresponding predictors poses a hard combinatorial optimization problem; as the following theorem shows, the problem is NP-hard even in the simple case where the set of predictor attributes to be used for each attribute is fixed, as expressed in the following theorem.

(8) Please replace paragraph 75 with the following paragraph showing changes.

Given the inherent difficulty of the CaRT-based semantic compression problem, the CaRTSelector 120 ~~130~~ implements two distinct heuristic search strategies that employ the Bayesian network model of T built by the DependencyFinder 110 to intelligently guide the search through the huge space of possible attribute prediction alternatives. The first strategy is a simple "greedy" selection algorithm that chooses CaRT predictors greedily based on their storage benefits during a single "roots-to-leaves" traversal of the Bayesian graph. The second, more complex strategy takes a less myopic approach that exploits the similarities between the CaRT-selection problem and WMIS; a key idea here is to determine the set of predicted attributes (and the corresponding CaRTs) by obtaining (approximate) solutions to a number of WMIS instances created based on the Bayesian model of T.

(9) Please replace paragraph 130 with the following paragraph showing changes.

A critical input parameter to the compression algorithms is the error tolerance for numeric attribute X_i is specified as a percentage of the width of the range of X_i-values in the table. Another important parameter to the table compressor 100 is the size of the sample that is used to select the CaRT models in the final compressed table. For these two parameters, the default values of 1% (for error tolerance) and 50KB (for sample size), respectively, are used in all of the referenced experiments. Note that 50KB corresponds to 0.065% to 0.475% and .174% of the total size of the Forest-cover, Corel and Census data sets, respectively. Finally, unless stated ~~states~~ otherwise, the table compressor 100 always uses MaxIndependentSet for CaRT-selection and the integrated pruning and building algorithm for constructing regression trees.

(10) Please replace paragraph 133 with the following paragraph showing changes.

Another crucial difference between fascicle and CaRT-based compression is that, when fascicles are used for compression, each tuple and as a consequence, every attribute value of a tuple is assigned to a single fascicle. However, in the table compressor 100, a predictor attribute and thus a predictor attribute value (belonging to a specific tuple) can be used in a number of different~~,~~ CaRTs to infer values for multiple different predicted attributes. Thus, CaRTs offer a more powerful and flexible model for capturing attribute correlations than fascicles.

(11) Please replace paragraph 135 with the following paragraph showing changes.

The compression ratios for the table compressor 100 are even more impressive for larger values of error tolerance (e.g., 10%) since the storage overhead of CARTs + outliers is even smaller at these higher error values. For example, at 10% error, in the compressed Corel data set, CaRTs consumer only 0.6 MB or 5.73% of the original table size. Similarly, for Forest-cover, the CaRT storage overhead reduces to 2.84 MB or 3.72% of the uncompressed table. The only exception is the Census data set where the decrease ~~decreased~~ in storage overhead is much steeper for fascicles than for CaRTs. One possible reason for the preceding is because of the small attribute domains in the Census data that cause each fascicle to cover a large number of tuples at higher error threshold values.

(12) Please replace paragraph 139 with the following paragraph showing changes.

With respect to running times, it was found that, in general, the MaxIndependentSet with parents performs quite well across all the data sets. This may be because the MaxIndependentSet constructs few CaRTs (118 for Census, 32 for Corel, and 46 for Forest-cover) and since it restricts the neighborhood for each attribute to only its parents, each CaRT ~~CaRt~~ contains few predictor attributes. While greedy does build the fewest CaRTS in most cases (10 for Census, 16 for Corel, and 19 for Forest-cover), all the materialized ancestors of an attribute are used as predictor attributes when building the CaRT for the attribute. As a result, since close to 50% attributes are materialized for the data sets, each CaRT is built using a large number of attributes, thus hurting greedy's performance.